Acadgild

Full-stack Web Development

Acadgild

Full-stack Web Development

Agenda

1. What is JavaScript?
2. Why JavaScript
3. JavaScript Evolution
4. What can we do with JavaScript?
5. Understanding JavaScript Code
6. How to run JavaScript
7. JavaScript Code Placement
8. Introduction to DevTools
9. Logging to Console
10. Variables
11. Identifiers
12. Reserved Words

Agenda

13 Literals

14 Data Types

15 Loosely Typed Language

Comments

16 Expressions & Operators

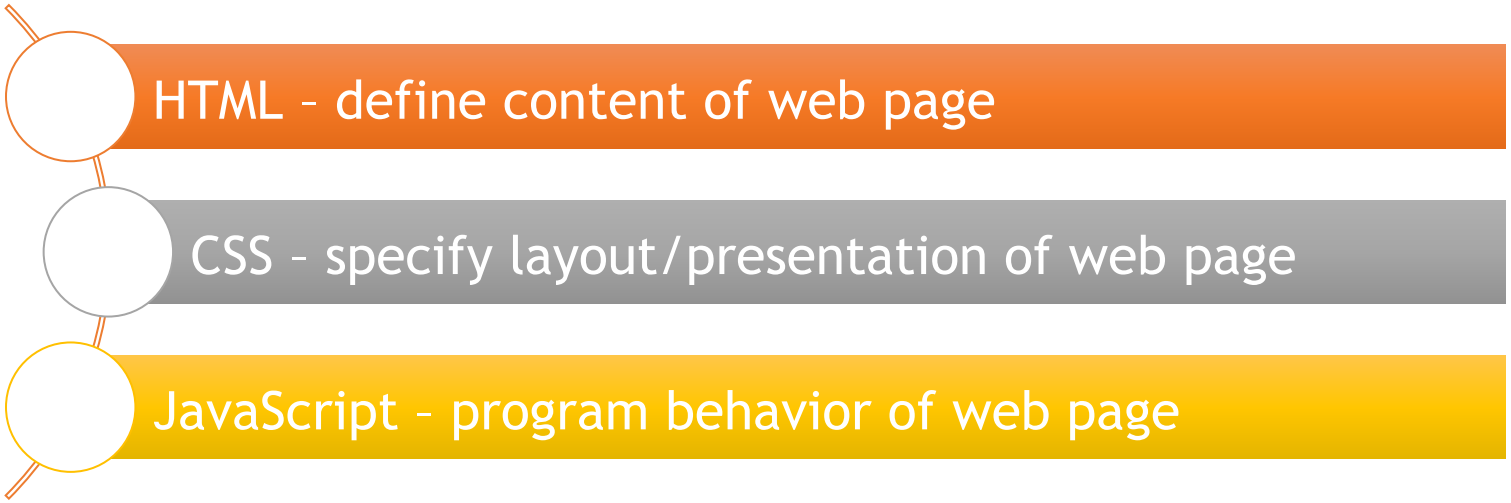
17 Operator Precedence

18

What is JavaScript?

JavaScript is the programming language of HTML and the Web.

It is one of the **3 languages** of the Web



Client side scripting language

An interpreted programming language with object-oriented capabilities

Why JavaScript

- Dynamic and lightweight
- Only programming language for web browsers
- Open Source and cross-platform

JavaScript Evolution

- Developed by Netscape in 1995 to create applications that run in the browser
- Brendan Eich is the father of JavaScript
- ES5 is most supported version in browsers
- ES6 or ES2015 is significant update in history of JavaScript
- JavaScript Today
  - ES8 or ES2017

Uniform distribution

Sample space $S = \{1, 2, 3, \ldots, k\}$.

Probability measure: equal assignment $(1/k)$ to all outcomes, *ie* all outcomes are equally likely.

Random variable X defined by $X(i) = i$, $(i = 1, 2, 3, \ldots, k)$.

Distribution: $P(X = x) = \frac{1}{k}$ $\quad (x = 1, 2, 3, \ldots, k)$

Moments:

$$\mu = E[X] = \frac{(1 + 2 + \cdots + k)}{k} = \frac{\frac{1}{2}k(k+1)}{k} = \frac{k+1}{2}$$

$$E[X^2] = \frac{(1^2 + 2^2 + \cdots + k^2)}{k} = \frac{\frac{1}{6}k(k+1)(2k+1)}{k} = \frac{(k+1)(2k+1)}{6}$$

$$\Rightarrow \sigma^2 = \frac{k^2 - 1}{12}$$

For example, if X is the score on a fair die, $P(X = x) = 1/6$ for $x = 1, 2, \ldots, 6$.

What Can We Do With JavaScript

- Change HTML content programmatically
- Develop dynamic and interactive web applications
- Games in the browser
- Develop mobile applications

Understanding JavaScript Code

- Is a list of programming **statements**
- Statements are composed of Values, Operators, Expressions, Keywords, and Comments
- Statements are generally followed by a semicolon character
- Statements are executed, one by one, in the same order as they are written
- JavaScript is case sensitive

How to run JavaScript

- JavaScript runs inside a browser on a program called JavaScript virtual machine or JavaScript engine
- JS code must be inserted between <script> tag in HTML file

```
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="UTF-8">
      <title>Hello World JS</title>
    </head>
    <body>
      <h2>Say Hello !!!</h2>
      <p id="message"></p>
      <script>
         alert("Hello World from JavaScript");
      </script>
    </body>
</html>
```

JavaScript Code Placement

➢ You can place JavaScript inside HTML file in following ways

1. **<script>** tag inside **<head>...</head>** section
2. **<script>** tag inside **<body>...</body>** section
3. **<script>** tag inside **<body>...</body>** and **<head>...</head>** sections
4. **<script>** tag referring an external file via **src** attribute and then include in **<head>...</head>** section (Most preferred way)

DevTools

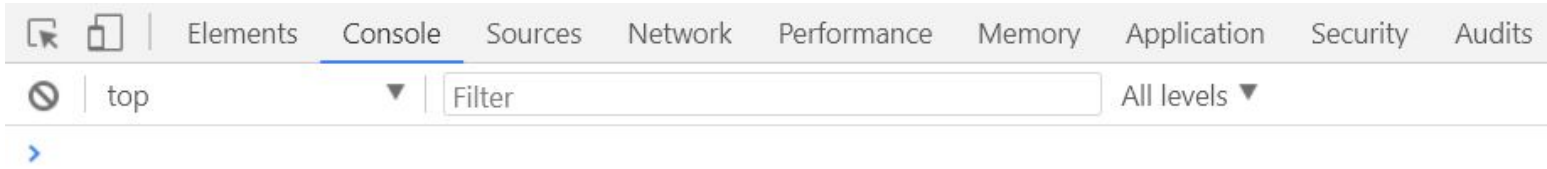
Developer tools are a collection of software that makes life easier for developers

All popular browsers has support for DevTools

Below screenshot is an example of chrome devtools

Console tab is JavaScript REPL(Read Evaluate Print Loop), where you can quickly test your JS code

Sources tab – browse and debug your JS code

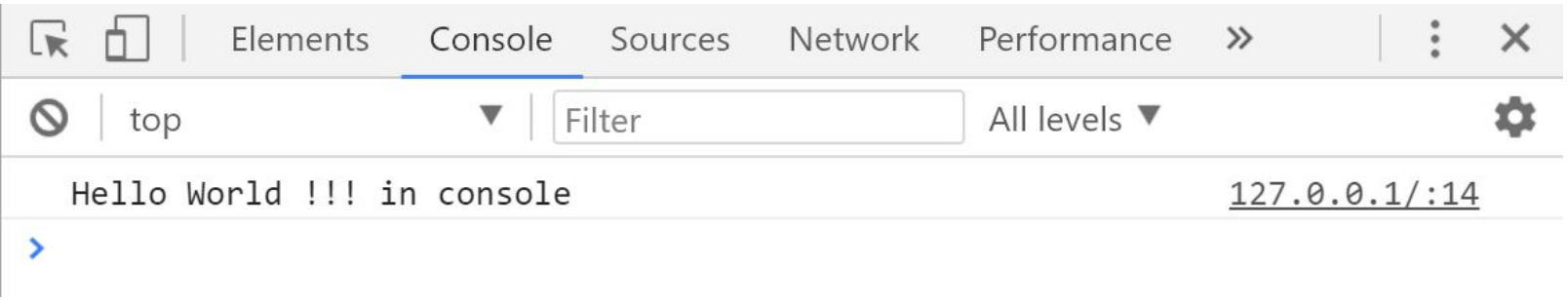


Logging to Console

- JavaScript provides a built-in feature to log messages to JavaScript console
- Used for debugging purpose
- Use console.log() function

```
<script>
console.log('Hello World !!! in console')
</script>
```

- You see below log in JavaScript console



Variables

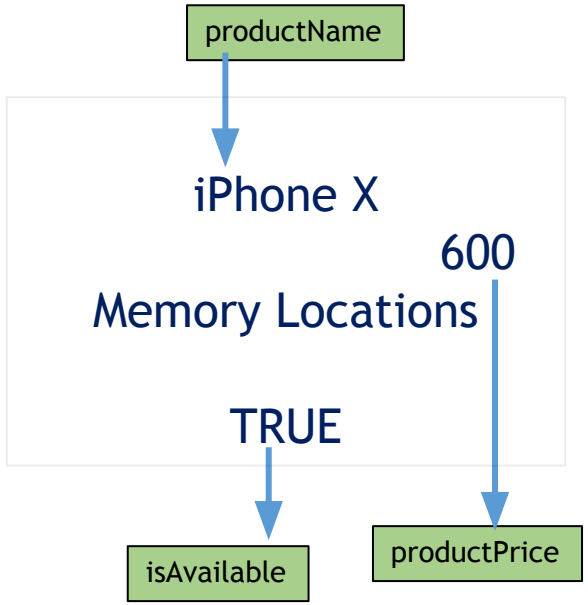
Variables are named containers or named memory locations which contain program data

Value of a variable can change over time

Before you use a variable, you must declare it by using **var** keyword

Storing a value in a variable for first time is called **variable initialization**

```
<script>
var productName = 'iPhone X'; // declaration &
initialization at same time
var isAvailable = true;
var productPrice; // contains "undefined" on declaration
// initialization
productPrice = 800;
</script>
```



Identifier Names

Variable names are called identifiers, they have naming rules

- Should not start with a numeral (0-9) and must begin with a letter, an underscore character (_) or dollar sign ($)
- Cannot be a reserved word
- Case-sensitive

Reserved Words

The following words can't be used as an identifier (the name of a variable, property, or function) in JavaScript:

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	FALSE	native	throws
catch	final	new	transient
char	finally	null	TRUE
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

Literals

- The way you represent values in JavaScript
- These are fixed values that you *literally* provide in your application source, and are not variables

```
'JavaScript is amazing' // string literal
300 //numeric literal
true // boolean literal
{ username: 'superman', password : 'joker' } // object literal
```

Data Types

- In real world we deal with so many different types of data. For example an product can have kinds of data as below
  - Name – string
  - Weight – number
  - Price – number
  - Manufactured date – Date
- Data types defines rules for the kind of data
- Data types are classified into 2 categories
  - Primitive Types
    - predefined types of data
    - immutable
  - Reference Types
    - store references to their data

Data Types (cont.)

Primitive Types

1. String
2. Number
3. Boolean
4. Undefined
5. Null
6. Symbol (ES6)

Reference Types

7. Object

string

- A string is a sequence of characters
- Strings are written within quotes.
- You can use either single or double quotes to represent string

```
var message = 'JavaScript is amazing'
var quote = "Pratice makes man perfect"
var multipleLanguages = "中文 español deutsch English हिन्दी العربية "
```

Number

Javascript only has one numeric data type which represents all types of numbers

JavaScript recognizes four types of numeric data:

- Decimal
- Binary
- Octal
- Hexadecimal

```
//numbers
var age = 20; // integer
var tempetaure = 21.5; // decimal
var color = 0x0000ff; // hexadecimal
var umask = 0o32; // octal
var result = 4.0e6; // exponential (4.0 × 10^6)
var nan = NaN; // "not a number"
```

Boolean

Booleans types have only two possible values: **true** and **false**

```
//boolean
var qualified = true;
var inProgress = false;
```

null and undefined

null and **undefined** are 2 special types in JavaScript

These represent something that doesn't exist

undefined means a variable has been declared but the value of that variable has not yet been defined

null is an assignment value. It can be assigned to a variable as a representation of no value

Since both types represent similar purpose it is the most confusing concept in JavaScript

```
/*
undefined is implicit when no value is initialized
*/
var total;

/* if a programmer needs to represent non-existence of value then assign null to a
variable */
total = null;
```

Object

- Primitives represent single value where as objects can represent multiple values
- In simple terms object is a container of primitives or objects
- Syntax of object : curly braces ({ and }).

You will learn more about objects in future modules

```
var user = { username: 'superman', password : 'joker' }
```

Loosely typed

- JavaScript is a **loosely typed** language, **which means you do not declare the data types of variables explicitly**

- Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned or re-assigned values of all types

```
var age = 42; // age is now a Number
var age = 'forty two'; // age is now a String
var age = true; // age is now a Boolean
```

Comments

- You can escape code from execution with comments
- Comments are helpful to explain or document your views / assumptions about code
- JS provides following ways of comment:
  - Single line
    - Starts with //
  - Multi line
    - start with /* and end with */

```
//null and undefined – single line comment

/*
undefined is implicit
when no value is initialized – multi line comment
*/
```

Expressions, Operators & Operator Types

- **Expression** is a statement that **evaluates to a value**
- **Operators** are like verb to an **expression**
- An expression is made of operators and operands
- JavaScript operators are classified based on kind of operation they provide
- JavaScript support the following types of operators
  - Arithmetic Operators
  - Comparison Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - String operators
  - Typeof operator

Arithmetic

Arithmetic operators perform arithmetic on numbers (literals or variables)

(Assume variable x holds 10 and variable y holds 5 for below example explanation)

Operator	Description	Example
+	Addition, adds two operands	x + y will give 15
-	Subtraction, Subtracts the second operand from the first	x - y will give 5
*	Multiplication, Multiply both operands	x * y will give 50
/	Division, Divide the numerator by the denominator	x / y will give 2
%	Modulus (Remainder), Outputs the remainder of an integer division	x % y will give 0
++	Increment, Increases an integer value by one	x++ will give 11
--	Decrement, Decreases an integer value by one	x-- will give 9

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.
(Assume variable x holds 10 and variable y holds 5 for below example explanation)

Operator	Description	Example
==	(equal) Checks if the value of two operands are equal or not, if yes, then the condition becomes true	x == y is not true
===	(strict equal) Checks if the value and type of two operands are equal or not, if yes, then the condition becomes true	x === y is not true
!=	(not equal) Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true.	x != y is true
!==	not equal value or not equal type	x !== y is true
>	(greater than) Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.	x > y is true
<	(less than) Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true.	x < y is false
>=	(greater than or equal to) Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.	x >= y is true
<=	(less than or equal to) Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.	x <= y is false
?:	ternary operator, If Condition is true? Then value x : Otherwise value y	**10 <= 8 ? "Ten is great" : " 8 is great"** results in "8 is great"

Logical operators

Logical operators are used to determine the logic between variables or values

(Assume variable x holds 10 and variable y holds 5 for below example explanation)

Operator	Description	Example
&&	(logical and) If both the operands are non-zero, then the condition becomes true.	x > 10 && y < 5 will evaluate to true
||	(logical or) If any of the two operands are non-zero, then the condition becomes true.	x > 10 || y > 15 will evaluate to true
!	(logical not) Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.	!(x > 10) will evaluate false (x > 10 is true and negation of true is false)

Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Description	Example
=	(Simple Assignment) Assigns values from the right side operand to the left side operand	x = y, value of y is assigned to x
+=	(Add and Assignment) It adds the right operand to the left operand and assigns the result to the left operand.	x += y, x and y are added and the result is assigned back to x
-=	(Subtract and Assignment) It subtracts the right operand from the left operand and assigns the result to the left operand.	x -= y, y is subtracted from x and the result is assigned back to x
*=	(Multiply and Assignment) It multiplies the right operand with the left operand and assigns the result to the left operand.	x *= y, x is mutiplied by y and the result is assigned back to x
/=	(Divide and Assignment) It divides the left operand with the right operand and assigns the result to the left operand	x /= y, x is divided by y and the result is assigned back to x
%=	(Modulus and Assignment) It takes modulus using two operands and assigns the result to the left operand	x %= y, x is divided by y and the remainder result is assigned back to x

String Operators

The + operator can also be used to concatenate strings.

```
var firstName = "Kim";
var secondName = "Keller";
var fullName = firstName + " " + secondName;
```

typeof operator

- Used to find the data type of variable
- Returns type information as a string
- six possible values that **typeof** returns
  - "number"
  - "string"
  - "boolean"
  - "object"
  - "function" and
  - "undefined"

Operator Precedence

It is very important to understand precedence of operators as their evaluation will result in different computations

Operators with higher precedence will be evaluated first and computation continues with next other higher precedence

Refer the below link for all JS operator precedence

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

Email us - support@acadgild.com